Amendments to the Claims:

This listing of claims will replace prior versions, and listings of claims in the application.

1. (Currently amended) A method for accessing status information related to a process ~~that is executable by one or more nodes from over a network~~, the method comprising:

~~retrieving status information related to an executable process by a process management system executing on a primary node; and~~

~~storing status information related to the executable process into a data structure, wherein the data structure is available to any node capable of accessing the process management system.~~

receiving a request from a client for status information related to the process;

identifying nodes in a network, each of the nodes executing a distributed thread of the process;

polling each identified node for status information associated with the thread executing by the node, the status information generated by a script associated with the process;

receiving the status information from each of the nodes;

storing the status information in a data structure; and

enabling the client to access the status information.

2. (Canceled)

3. (Currently amended) The method of claim 1, further comprising:

invoking one or more script engines ~~by the process management system~~ to execute at least one script code that performs ~~the~~ at least one action of the ~~executable~~ process~~,~~ ;

~~wherein the process management system handles~~ handling multiple script threads during the execution of the process.

4. (Currently amended) The method of claim 3, wherein the one or more script engines are maintained by a process management system that executes on the ~~one or more~~ nodes.

5. (Currently amended) The method of claim 4, wherein the one or more nodes include ~~the~~ a primary node.

6. (Currently amended) The method of claim 1, ~~wherein the step of retrieving includes polling the process management system on the one or more nodes to~~

~~obtain status information related to the executable process~~ further comprising making the data structure available to any node in the network capable of accessing a process management system in a primary node.

7. (Currently amended) The method of claim 6, wherein the step of polling is performed by the process management system residing on the primary node over an established connection with the ~~one or more~~ identified nodes.

8. (Currently amended) The method of claim 7, wherein the ~~one or more~~ identified nodes include the primary node.

9. (Canceled)

10. (Canceled)

11. (Canceled)

12. (Currently amended) The method of claim 1, wherein the step of storing is performed by ~~the~~ a process management system executing on ~~the~~ a primary node.

13. (Currently amended) The method of claim 12, wherein the step of storing further includes:

placing the status information relative to the executable process into a private data structure by the process management system on the primary node, wherein the private data structure is accessible to only ~~the~~ script threads that are spawned during the execution of the process.

14. (Currently amended) The method of claim 12, wherein the step of storing further includes:

placing the status information relative to the executable process into a status value data structure that is accessible to any node capable of accessing the process management system executing on the primary node.

15. (Currently amended) The system of claim ~~16~~ 14, wherein the status value data structure comprises data for providing an indication of an event that occurs during the execution of the process.

16. (Currently amended) The method of claim 1, further comprising:

establishing a connection between a process management system executing on at least one of the ~~one or more~~ nodes and ~~the~~ another process management system residing on ~~the~~ a primary node, wherein the connection is established by ~~the~~ a script code in execution by the ~~one or more~~ a script engine~~s~~ associated with the at least one node.

17. (Currently amended) The method of claim 1, further comprising:

establishing a connection between ~~one or more nodes~~ other client nodes and ~~the~~ a process management system residing on ~~the~~ a primary node, wherein the connection is established from a user interface executing on the ~~one or more nodes~~ other client nodes; and

accessing the process management system from over the established connection by the user interface executing on the ~~one or more nodes~~ other client nodes.

18. (Original) The method of claim 17, wherein the step of establishing includes accepting a command as input by the user interface to establish a connection with the process management system executing on the primary node.

19. (Original) The method of claim 17, wherein the step of accessing includes accepting a command as input by the user interface to invoke the action of the executable process by the process management system from over the established connection.

20. (Original) The method of claim 17, wherein the step of accessing includes accepting a command as input by the user interface to poll the process management system for status information from over the established connection.

21. (Original) The method of claim 17, wherein the user interface receives messages from the process management system over the established connection.

22. (Original) The method of claim 21, wherein the messages contain information that is descriptive of the primary node.

23. (Original) The method of claim 21, wherein the messages contain information that is descriptive of a particular event that occurs during the execution of the process.

24. (Original) The method of claim 21, wherein the messages contain a data structure that is generated as a result of the execution of the script code by the one or more script engines to indicate the status of the executable process.

25. (Withdrawn) A computer-readable medium having computer executable components comprising:

a first component for hosting one or more script engines, and for managing script threads that are spawned during the execution of a process;

a second component for handling the execution and monitoring of local processes that are launched during the execution of script code by the one or more script engines;

a third component for:

(i) receiving and accepting requests from one or more nodes to establish a connection over a network;

(ii) receiving commands from the one or more nodes from over an established connection to invoke the action of the executable process;

(iii) receiving commands from the one or more nodes from over the established connection to request for information that is descriptive of the primary node; and

(iv) sending messages to the one or more nodes from over the established connection in response to the requests and commands received from the one or more nodes.

a fourth component for:

(i) enabling messages to be passed between the first component and the third component;

(ii) launching the execution of a primary script file on behalf of the first component, wherein the primary script file spawns the execution of the one or

more script engines in response to a request received over the established connection to invoke an executable process;

(iii) storing information related to one or more script threads and local processes in execution; and

(iv) storing status information relative to the executable process into a data structure that is accessible to the script threads.

26. (Withdrawn) The computer-readable medium of claim 25, wherein the first component further comprises: a primary script file for spawning the execution of any of the one or more script engines in response to a request to invoke the executable process, the request indicating a specific script file to execute..

27. (Withdrawn) The computer-readable medium of claim 26, wherein the primary script file is automatically loaded onto the primary node for execution when the process management system is initiated.

28. (Withdrawn) The computer-readable medium of claim 26, wherein a failure to load the primary script file onto the primary node results in the first component indicating failure and terminating the established connection.

29. (Withdrawn) The computer-readable medium of claim 26, wherein the request is made by one or more nodes over an established connection.

30. (Withdrawn) The computer-readable medium of claim 26, wherein the request is made by a local process over an established connection.

31. (Withdrawn) The computer-readable medium of claim 25, wherein the one or more script engines hosted by the first component terminate the executable process when an execution error occurs.

32. (Withdrawn) The computer-readable medium of claim 25, wherein the first component further comprises: a global object for extending the behavior of the one or more script engines, the script engine obtaining an identifier from the global object to perform a specific task upon encountering the identifier during the execution of the script code and determining that it cannot be interpreted.

33. (Withdrawn) The computer-readable medium of claim 325, wherein the identifier is an executable script variable having instructions for performing a specific task.

34. (Withdrawn) The computer-readable medium of claim 32, wherein the global object implements a communication channel between the first component and the one or more script engines to enable the one or more script engines to interact with the first component.

35. (Withdrawn) The computer-readable medium of claim 32, wherein the global object implements a communication channel between the first component and the one or more script engines to enable the first component to interact with the one or more script engines.

36. (Withdrawn) The computer-readable medium of claim 25, wherein the second component launches local processes that are called by a script in execution using the global object, and reports the status information and data generated by the local processes back to the calling script.

37. (Withdrawn) The computer-readable medium of claim 25, wherein the fourth component receives information from the second component that is descriptive of the local processes in execution by the one or more script engines.

38. (Withdrawn) The computer-readable medium of claim 25, wherein the fourth component receives status information from the one or more script threads and

stores the information into a public data structure that is accessible to the one or more nodes capable of establishing a connection with the first component.

39. (Withdrawn) The computer-readable medium of claim 25, wherein the fourth component receives status information from the one or more script threads and stores the information into a private data structure that is accessible to only the script threads in operation during process execution.

40. (Withdrawn) The computer-readable medium of claim 25, wherein the fourth component receives status information from the one or more script threads and stores the information into a status value data structure that is accessible to the one or more nodes capable of establishing a connection with the first component.

41. (Withdrawn) The computer-readable medium of claim 40, wherein the status value data structure contains data for providing an indication of an event that occurs during the execution of the process.

42. (Currently amended) A system ~~for accessing status information that is stored on a primary node, wherein the status information is related to a process that is executable by one or more nodes from over a network, the system~~ comprising:

~~one or more user interfaces for invoking the executable process and retrieving status information generated by one or more script engines in execution from over a network;~~

~~a multiple threaded process management system executing on a primary node for collecting and storing status information related to the executable process from the one or more nodes;~~

~~at least one script engine maintained by the process management system for accessing and executing script code; and~~

~~at least one database having stored therein script code for enabling the executable process, wherein the database is accessible by the at least one script engine~~

a process management system executing on a primary node in a network, the process management system configured to collect status information associated with a process, the processing management system also configured to divide the process into multiple threads and distribute the threads to multiple remote nodes in the network, the process management system further configured to receive the status information associated with the threads from each remote node and store the status information in a data structure accessible by any node with authorized access to the process management system; and

the remote nodes in the network, each remote node processing at least one of the threads associated with the process and including a script configured to provide the status information collected by the process management system.

43. (Currently amended) The system of claim 42, further comprising one or more client node each configured with a user-interface, ~~wherein~~ the one or more user interfaces configured to establish a connection over the network with the process management system executing on the primary node, the one or more user interfaces also configured to request the status information from the process management system and to process the status information when the information is received.

44. (Canceled)

45. (Canceled)

46. (Original) The system of claim 42, wherein the one or more user interfaces accept as input commands to establish a connection with the process management system executing on the primary node.

47. (Original) The method of claim 42, wherein the one or more user interfaces accept as input commands to invoke the action of the executable process by

the process management system, and sends requests to invoke the action of the executable process to the process management system from over the established connection.

48. (Currently amended) The ~~method~~ system of claim 42, wherein the one or more user interfaces accept as input commands to poll the process management system for status information, and sends requests to poll the process management system for status information from over the established connection.

49. (Original) The system of claim 42, wherein the one or more user interfaces receive messages from the process management system over the established connection in response to the polling.

50. (Currently amended) The ~~method~~ system of claim 49, wherein the messages contain information that is descriptive of the primary node.

51. (Currently amended) The ~~method~~ system of claim 49, wherein the messages contain information that is descriptive of a particular event that occurs during the execution of the process.

52. (Currently amended) The ~~method~~ system of claim 49, wherein the messages contain a data structure that is generated as a result of the execution of the script code by the one or more script engines to indicate the status of the executable process.

53. (Currently amended) The system of claim 42, wherein the process management system accepts connection requests from one or more user interfaces operating on one or more nodes associated with the process management system ~~from~~ over ~~the~~ an established connection.

54. (Original) The system of claim 53, wherein the one or more nodes include the primary node.

55. (Original) The system of claim 42, wherein the process management system receives requests to invoke the action of the executable process from the one or more nodes connected to the process management system.

56. (Original) The system of claim 42, wherein the process management system continuously polls the one or more nodes connected to the process management system to obtain status information related to the executable process.

57. (Currently amended) The system of claim 42, wherein the process management system stores the information into a public data structure that is accessible to the one or more nodes capable of establishing a connection with the ~~first component~~ process management system.

58. (Currently amended) The system of claim 42, wherein the process management system stores the status information relative to the process into a private data structure that is accessible to only ~~the~~ script threads in operation during process execution.

59. (Original) The system of claim 42, wherein the process management system stores the status information relative to the executable process into a status value data structure that is accessible to the one or more nodes having access to the status information.

60. (Original) The system of claim 59, wherein the status value data structure contains data for providing an indication of a particular event that occurs during the execution of the process.

61. (Original) The system of claim 42, wherein the process management system receives requests for status information relative to the executable process from the one or more nodes connected to the process management system.

62. (Original) The system of claim 42, wherein the process management system sends the public data structure to the one or more nodes in response to the request.

63. (Original) The system of claim 42, wherein the process management system sends the status value data structure to the one or more nodes in response to the request.

64. (New) An apparatus comprising:

means for receiving a request from a client to initiate a process;

means for dividing the process into multiple threads;

means for distributing the threads to multiple nodes in a network for execution;

means for polling each node for status information generated by a script executing in the node;

means for receiving the status information from each of the nodes;

means for storing the status information in a data structure; and

means for enabling any node with authorization to access the status information.